Subject: [PATCH 3/4] namespace containers: add nsproxy to nscont struct
Posted by serue on Mon, 19 Feb 2007 22:16:20 GMT
View Forum Message <> Reply to Message

From: Serge E. Hallyn <serue@us.ibm.com>
Subject: [PATCH 3/4] namespace containers: add nsproxy to nscont struct

Each ns container is associated with an nsproxy. Add that
nsproxy to the nscont struct, set it when a container is auto-
created on clone/unshare, and inc/dec the nsproxy to account for
each container referencing it.

Note that once the nscont->nsproxy is set, it will never change
for the duration of the container's lifetime.

Changelog:
Feb 14: added ss->init_from_task() hook so ns_container
can initialize a container's private data from a task
on clone().

Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>

Documentation/containers.txt | 9 +++++++++
include/linux/container.h | 1 +
include/linux/nsproxy.h | 1 +
kernel/container.c | 16 ++++++++++++++++
kernel/ns_container.c | 11 +++++++++++
5 files changed, 38 insertions(+), 0 deletions(-)

f863632142517f79ef885c238a8e5df238e8420c
diff --git a/Documentation/containers.txt b/Documentation/containers.txt
index 7918827..0001191 100644
--- a/Documentation/containers.txt
+++ b/Documentation/containers.txt
@@ -466,6 +466,15 @@ LL=manage_mutex
The container system is about to destroy the passed container; the
subsystem should do any necessary cleanup

+int init_from_task(struct container *cont, struct task_struct *task)
+LL=manage_mutex
+
+Called during a container_clone() call to allow differentiation
+between a container created automatically and one created by hand.
+A container created by hand inherits the nsproxy from the parent
+container. A container created automatically inherits the nsproxy
+from the task entering, which may have already done some unsharing.

```
+
 int can_attach(struct container_subsys *ss, struct container *cont,
        struct task_struct *task)
 LL=manage_mutex
diff --git a/include/linux/container.h b/include/linux/container.h
index db2fc27..4c9c092 100644
--- a/include/linux/container.h
+++ b/include/linux/container.h
@@ -197,6 +197,7 @@ struct container_subsys {
  int (*create)(struct container_subsys *ss,
        struct container *cont);
  void (*destroy)(struct container_subsys *ss, struct container *cont);
+ void (*init_from_task)(struct container *cont, struct task_struct *task);
  int (*can_attach)(struct container_subsys *ss,
      struct container *cont, struct task_struct *tsk);
  void (*attach)(struct container_subsys *ss, struct container *cont,
diff --git a/include/linux/nsproxy.h b/include/linux/nsproxy.h
index d11eb09..43f5696 100644
--- a/include/linux/nsproxy.h
+++ b/include/linux/nsproxy.h
@@ -71,6 +71,7 @@ static inline void swap_nsproxies(struct
  put_nsproxy(oldnsp);
 }

+struct container;
 #ifdef CONFIG_CONTAINER_NS
 int ns_container_clone(struct task_struct *tsk, struct nsproxy *nsproxy);
 #else
diff --git a/kernel/container.c b/kernel/container.c
index 0606753..0352f84 100644
--- a/kernel/container.c
+++ b/kernel/container.c
@@ -920,6 +920,19 @@ static int attach_task(struct container
  return 0;
 }

+static void init_container_from_task(struct container *cont,
+     struct task_struct *tsk)
+{
+ struct container_subsys *ss;
+ int h = cont->hierarchy;
+
+ for_each_subsys(h, ss) {
+  if (ss->init_from_task) {
+   ss->init_from_task(cont, tsk);
+  }
+ }
+}
```

```
+
 /*
  * Attach task with pid 'pid' to container 'cont'. Call with
  * manage_mutex, may take callback_mutex and task_lock of task
@@ -1665,6 +1678,9 @@ int container_clone(struct task_struct *
   goto again;
  }

+ /* the new container needs private info initialized from the task */
+ init_container_from_task(child, tsk);
+
  /* All seems fine. Finish by moving the task into the new container */
  ret = attach_task(child, tsk);
  mutex_unlock(&manage_mutex);
diff --git a/kernel/ns_container.c b/kernel/ns_container.c
index 23fac0e..1cc9cea 100644
--- a/kernel/ns_container.c
+++ b/kernel/ns_container.c
@@ -11,6 +11,7 @@

 struct nscont {
  struct container_subsys_state css;
+ struct nsproxy *nsproxy; /* never changes once set */
  spinlock_t lock;
 };

@@ -82,10 +83,19 @@ static int ns_create(struct container_su
  return 0;
 }

+void ns_init_from_task(struct container *cont, struct task_struct *tsk)
+{
+ struct nscont *ns = container_nscont(cont);
+ ns->nsproxy = tsk->nsproxy;
+ get_nsproxy(ns->nsproxy);
+}
+
 static void ns_destroy(struct container_subsys *ss,
         struct container *cont)
 {
  struct nscont *ns = container_nscont(cont);
+ if (ns->nsproxy)
+  put_nsproxy(ns->nsproxy);
  kfree(ns);
 }

@@ -97,6 +107,7 @@ static struct container_subsys ns_subsys
  //.attach = ns_attach,
```

```
  //.post_attach = ns_post_attach,
  //.populate = ns_populate,
+ .init_from_task = ns_init_from_task,
  .subsys_id = -1,
 };

--
1.1.6
```